Types in Logic Programming

Frank Pfenning
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
fp@cs.cmu.edu

Abstract

Types play an increasingly important role in logic programming, in language design as well as language implementation. We present various views of types, their connection, and their role within the logic programming paradigm.

Among the basic views of types we find the so-called descriptive systems, where types describe properties of untyped logic programs, and prescriptive systems, where types are essential to the meaning of programs. A typical application of descriptive types is the approximation of the meaning of a logic program as a subset of the Herbrand universe on which a predicate might be true. The value of prescriptive systems lies primarily in program development, for example, through early detection of errors in programs which manifest themselves as type inconsistencies, or as added documentation for the intended and legal use of predicates.

Central topics within these views are the problems of type inference and type reconstruction, respectively. Type inference is a form of analysis of untyped logic programs, while type reconstruction attempts to fill in some omitted type information in typed logic programs and generalizes the problem of type checking. Even though analogous problems arise in functional programming, algorithms addressing these problems are quite different in our setting.

Among the specific forms of types we discuss are simple types, recursive types, polymorphic types, and dependent types. We also briefly touch upon subtypes and inheritance, and the role of types in module systems for logic programming languages.